Remotion SaaS Motion Graphics: Best Practices & Code Snippets

A comprehensive guide to creating exceptional, production-ready motion graphics for SaaS products using Remotion. Based on industry best practices and top-tier examples like Calendly, RingCentral, Superhuman, and Loom.

---

1. Architecture & Project Structure

Optimal Folder Organization

```
saas-video/
├── src/
│ ├── Root.tsx # Composition registry
│ ├── components/
│ │ ├── scenes/
│ │ │ ├── IntroScene.tsx # Opening title sequence
│ │ │ ├── FeatureScene.tsx # Feature demonstrations
│ │ │ └── CTAScene.tsx # Call-to-action
│ │ ├── common/
│ │ │ ├── AnimatedText.tsx # Reusable text animations
│ │ │ ├── Transition.tsx # Scene transitions
│ │ │ └── Background.tsx # Brand backgrounds
│ │ └── effects/
│ │ ├── GlassEffect.tsx # Glassmorphism
│ │ ├── ParticleField.tsx # Particle systems
│ │ └── GradientShift.tsx # Dynamic gradients
│ ├── styles/
│ │ ├── colors.ts # Brand color system
│ │ └── constants.ts # Timing, spacing, sizing
│ ├── data/
│ │ └── compositions.ts # Composition metadata
│ └── utils/
│ ├── animations.ts # Shared animation logic
│ └── text-animation.ts # Typography animations
├── public/
│ ├── assets/
│ │ ├── logo.svg
│ │ ├── icons/
│ │ └── backgrounds/
│ └── fonts/
├── remotion.config.ts
└── package.json
```

Root.tsx: Composition Registry Pattern

Best Practice: Use a metadata-driven approach for scalability.

```
import React from "react";
import { Composition } from "remotion";
import { IntroScene } from "./components/scenes/IntroScene";
import { FeatureScene } from "./components/scenes/FeatureScene";
import { CTAScene } from "./components/scenes/CTAScene";

export const RemotionRoot: React.FC = () => {
return (
<>
{/* 60-second SaaS explainer video */}
<Composition
id="SaaS_Explainer_60s"
component={SaaS_Explainer}
durationInFrames={1800} // 60 seconds @ 30fps
fps={30}
width={1920}
height={1080}
defaultProps={{
productName: "Your SaaS",
accentColor: "#6366F1",
darkMode: false,
}}
/>
```

```
  {/* Individual scene tests */}
  <Composition
    id="Intro_Test"
    component={IntroScene}
    durationInFrames={300} // 10 seconds
    fps={30}
    width={1920}
    height={1080}
  />

  <Composition
    id="Feature_Demo"
    component={FeatureScene}
    durationInFrames={600} // 20 seconds
    fps={30}
    width={1920}
    height={1080}
    />
  </>
```

```
);
};

// Main composition: Full video structure
const SaaS_Explainer: React.FC<{
productName: string;
accentColor: string;
darkMode: boolean;
}> = ({ productName, accentColor, darkMode }) => {
return (
<>
{/* Scene 1: Intro (0-10s) */}
```

```
    {/* Scene 2: Problem Statement (10-25s) */}
    <Sequence from={300} durationInFrames={450}>
      <FeatureScene
        title="Your Problem"
        description="Teams waste hours on manual processes"
        color={accentColor}
      />
    </Sequence>

    {/* Scene 3: Solution (25-45s) */}
    <Sequence from={750} durationInFrames={600}>
      <FeatureScene
        title="Our Solution"
        description="Automate workflows in minutes"
        color={accentColor}
      />
    </Sequence>

    {/* Scene 4: CTA (45-60s) */}
    <Sequence from={1350} durationInFrames={450}>
      <CTAScene productName={productName} color={accentColor} />
    </Sequence>
  </>
```

```
);
};
```

---

2. Animation Fundamentals for SaaS

2.1 Core Animation Concepts

The Three Pillars of SaaS Motion Graphics:

Concept	Purpose	Use Case
Spring Animation	Physics-based, natural feel	UI element bounces, icon reveals
Interpolate + **Easing**	Smooth linear transitions	Text slides, opacity fades, size scaling
Sequencing	Layered timing for storytelling	Scene transitions, staggered reveals

2.2 Spring Animation: The Foundation

Spring animations create natural, bouncy motion that feels premium and responsive—critical for SaaS perception.

```
import { spring, useCurrentFrame, useVideoConfig, interpolate } from "remotion";

export const SpringAnimationExample: React.FC = () => {
const frame = useCurrentFrame();
const { fps } = useVideoConfig();

// Create spring with physics parameters
const springValue = spring({
frame,
fps,
config: {
damping: 5, // Lower = more bouncy (1-50)
mass: 1, // Object "weight" (0.1-10)
stiffness: 100, // Spring tension (0.1-300)
overshootClamping: false, // Allow overshoot
},
delay: 0, // Frames to wait before animating
});

// Map spring (0->1) to pixel values (0->300)
const xPosition = interpolate(springValue, [0, 1], [0, 300]);

// Map spring to scale (0->1 becomes 0->1.2 for "pop" effect)
const scale = interpolate(springValue, [0, 1], [0, 1.2], {
```

```
extrapolateRight: "clamp",
});

return (
<div style={{ transform: translateX(${xPosition}px) scale(${scale}) }}>
▯ Bouncy Element
</div>
);
};

// BEST PRACTICE: Premade spring configs for consistency
export const SPRING_CONFIGS = {
fast: { damping: 8, mass: 0.5, stiffness: 150 }, // Quick UI reveals
smooth: { damping: 8, mass: 1, stiffness: 100 }, // Standard animations
sluggish: { damping: 12, mass: 1.5, stiffness: 50 }, // Dramatic elements
bouncy: { damping: 4, mass: 1, stiffness: 150 }, // Playful interactions
};
```

2.3 Interpolate + Easing: Precise Control

For predictable, linear animations (text slides, fades, scale transforms).

```
import { interpolate, Easing, useCurrentFrame } from "remotion";

export const InterpolateWithEasing: React.FC = () => {
const frame = useCurrentFrame();

// EXAMPLE 1: Simple fade-in (0-30 frames, opacity 0->1)
const fadeOpacity = interpolate(frame, [0, 30], [0, 1], {
extrapolateRight: "clamp",
});

// EXAMPLE 2: Slide in with easing curve
const slideX = interpolate(frame, [10, 60], [-100, 0], {
easing: Easing.out(Easing.cubic),
extrapolateRight: "clamp",
});

// EXAMPLE 3: Scale with overshoot (feels premium)
const scale = interpolate(frame, [0, 30], [0.5, 1], {
easing: Easing.out(Easing.elastic(1.2)), // Elastic overshoot
extrapolateRight: "clamp",
});

// EXAMPLE 4: Multi-point interpolation (complex sequences)
const complexValue = interpolate(
frame,
[0, 30, 60, 90], // Frame keypoints
[0, 1, 0.5, 1], // Output values
{
easing: Easing.inOut(Easing.quad),
extrapolateRight: "clamp",
}
);

return (
<div
style={{
opacity: fadeOpacity,
transform: translateX(${slideX}px) scale(${scale}),
}}
>
Animated Text
</div>
);
};

// EASING LIBRARY FOR SAAS (Professional feel)
export const EASING_PRESETS = {
// Entrances (0 -> 1)
fadeIn: Easing.out(Easing.quad),
slideInSmooth: Easing.out(Easing.cubic),
popIn: Easing.out(Easing.elastic(1.1)),

// Exits (1 -> 0)
fadeOut: Easing.in(Easing.quad),
slideOutSmooth: Easing.in(Easing.cubic),

// Attention getters
emphasis: Easing.inOut(Easing.elastic(1.3)),
pulse: Easing.inOut(Easing.sin),
};
```

2.4 Staggered Sequential Animations

Essential for revealing multiple list items, features, or benefits with professional timing.

```
import { interpolate, Easing, useCurrentFrame } from "remotion";

interface StaggeredListProps {
items: string[];
staggerDelay: number; // Frames between each item
startFrame: number;
itemDuration: number; // Frames per item animation
}

export const StaggeredFeatureList: React.FC<StaggeredListProps> = ({
items,
staggerDelay = 15,
startFrame = 0,
itemDuration = 40,
}) => {
const frame = useCurrentFrame();
```

return (
<div style={{ display: "flex", flexDirection: "column", gap: "20px" }}>
{items.map((item, index) => {
// Calculate per-item animation window
const itemStartFrame = startFrame + index * staggerDelay;
const itemEndFrame = itemStartFrame + itemDuration;

```
    // Calculate opacity for this item
    const opacity = interpolate(
      frame,
      [itemStartFrame, itemEndFrame],
      [0, 1],
      {
        easing: Easing.out(Easing.quad),
        extrapolateLeft: "clamp",
        extrapolateRight: "clamp",
      }
    );

    // Calculate slide position
    const slideX = interpolate(
      frame,
      [itemStartFrame, itemEndFrame],
      [-50, 0],
      {
        easing: Easing.out(Easing.cubic),
        extrapolateLeft: "clamp",
        extrapolateRight: "clamp",
      }
    );

    return (
      <div
        key={index}
        style={{
          opacity,
          transform: `translateX(${slideX}px)`,
          fontFamily: "Helvetica Neue, sans-serif",
          fontSize: "32px",
          fontWeight: "600",
          color: "#1F2937",
          transition: "none",
        }}
      >
        ✓ {item}
      </div>
    );
  })}
</div>
```

);
};

// Usage in composition
<StaggeredFeatureList
items={[
"Automate repetitive tasks",
"Save 10+ hours per week",
"Integrate with your workflow",
]}
staggerDelay={20}
startFrame={100}
itemDuration={50}
/>

---

3. Pro Techniques: SaaS-Specific Motion Patterns

3.1 Animated Text Components

Text is the hero of SaaS videos. Perfect animations build credibility.

import { useCurrentFrame, interpolate, Easing } from "remotion";

interface AnimatedHeadingProps {
text: string;
startFrame: number;
duration: number;
color?: string;
fontSize?: number;
}

export const AnimatedHeading: React.FC<AnimatedHeadingProps> = ({
text,
startFrame,
duration,
color = "#000",

```
fontSize = 72,
}) => {
const frame = useCurrentFrame();

// Fade in
const opacity = interpolate(
frame,
[startFrame, startFrame + duration * 0.3],
[0, 1],
{
easing: Easing.out(Easing.quad),
extrapolateRight: "clamp",
}
);

// Slide up
const translateY = interpolate(
frame,
[startFrame, startFrame + duration * 0.3],
[30, 0],
{
easing: Easing.out(Easing.cubic),
extrapolateRight: "clamp",
}
);

// Scale (subtle pop)
const scale = interpolate(
frame,
[startFrame, startFrame + duration * 0.2],
[0.95, 1],
{
easing: Easing.out(Easing.elastic(1.05)),
extrapolateRight: "clamp",
}
);

return (
<h1
style={{
fontSize,
fontWeight: "700",
color,
opacity,
transform: translateY(${translateY}px) scale(${scale}),
margin: 0,
fontFamily: "'Inter', sans-serif",
letterSpacing: "-1px",
}}
>
{text}
</h1>
```

```
);
};
```

```
// Character-by-character reveal (premium feel)
export const CharacterReveal: React.FC<AnimatedHeadingProps> = ({
text,
startFrame,
duration,
color = "#000",
}) => {
const frame = useCurrentFrame();
const chars = text.split("");
const charsPerFrame = chars.length / duration;
```

```
return (
<div style={{ display: "flex", gap: "2px", fontWeight: "700" }}>
{chars.map((char, i) => {
const charStartFrame = startFrame + i / charsPerFrame;
const charOpacity = interpolate(
frame,
[charStartFrame, charStartFrame + 5],
[0, 1],
{
easing: Easing.out(Easing.quad),
extrapolateLeft: "clamp",
extrapolateRight: "clamp",
}
);
```

```
      return (
        <span key={i} style={{ opacity: charOpacity, color }}>
          {char}
        </span>
      );
    })}
  </div>
```

```
);
};
```

3.2 UI Animation Patterns

Showcase UI interactions with professional animations.

```
import { AbsoluteFill, interpolate, Easing, useCurrentFrame } from "remotion";
```

```
// PATTERN 1: Button Click Animation
export const ButtonClickAnimation: React.FC = () => {
```

```
const frame = useCurrentFrame();

// Press down (0-5 frames)
const scalePress = interpolate(
frame,
[0, 5],
[1, 0.95],
{
easing: Easing.in(Easing.quad),
extrapolateRight: "clamp",
}
);

// Spring back (5-15 frames)
const scaleRelease = interpolate(
frame,
[5, 15],
[0.95, 1.05],
{
easing: Easing.out(Easing.elastic(1.1)),
extrapolateRight: "clamp",
}
);

const scale = frame < 5 ? scalePress : scaleRelease;

// Ripple effect
const rippleRadius = interpolate(frame, [0, 20], [0, 150], {
easing: Easing.out(Easing.quad),
});

const rippleOpacity = interpolate(frame, [15, 25], [0.5, 0], {
extrapolateRight: "clamp",
});

return (
<AbsoluteFill style={{ justifyContent: "center", alignItems: "center" }}>
<div
style={{
width: "200px",
height: "50px",
backgroundColor: "#6366F1",
borderRadius: "8px",
display: "flex",
alignItems: "center",
justifyContent: "center",
color: "white",
fontWeight: "600",
fontSize: "16px",
transform: scale(${scale}),
boxShadow: "0 4px 12px rgba(99, 102, 241, 0.3)",
position: "relative",
overflow: "hidden",
}}
>
Click Me
{/* Ripple effect */}
<div
style={{
position: "absolute",
width: rippleRadius * 2,
height: rippleRadius * 2,
backgroundColor: "rgba(255, 255, 255, 0.3)",
borderRadius: "50%",
top: "50%",
left: "50%",
transform: "translate(-50%, -50%)",
opacity: rippleOpacity,
}}
/>
</div>
</AbsoluteFill>
);
};

// PATTERN 2: Loading Bar Animation
export const LoadingBarAnimation: React.FC = () => {
const frame = useCurrentFrame();

// Smooth progress from 0 to 100% over 300 frames
const progress = interpolate(frame, [0, 300], [0, 100], {
easing: Easing.inOut(Easing.quad),
extrapolateRight: "clamp",
});

// Shimmer effect (optional premium touch)
const shimmerPosition = interpolate(
frame,
[0, 60],
[-300, 400],
{
easing: Easing.linear,
}
);

return (
<div
style={{
width: "400px",
height: "8px",
backgroundColor: "#E5E7EB",
borderRadius: "4px",
overflow: "hidden",
}}
```

```
>
<div
style={{
width: ${progress}%,
height: "100%",
backgroundColor: "#6366F1",
borderRadius: "4px",
position: "relative",
overflow: "hidden",
}}
>
{/* Shimmer */}
<div
style={{
position: "absolute",
top: 0,
left: shimmerPosition,
width: "100px",
height: "100%",
background:
"linear-gradient(90deg, transparent, rgba(255,255,255,0.3), transparent)",
}}
/>
</div>
</div>
);
};

// PATTERN 3: Modal Pop-In (High polish)
export const ModalPopIn: React.FC = () => {
const frame = useCurrentFrame();

// Backdrop fade
const backdropOpacity = interpolate(frame, [0, 20], [0, 0.5], {
easing: Easing.out(Easing.quad),
extrapolateRight: "clamp",
});

// Modal scale + translate
const scale = interpolate(frame, [0, 25], [0.7, 1], {
easing: Easing.out(Easing.elastic(1.2)),
extrapolateRight: "clamp",
});

const translateY = interpolate(frame, [0, 25], [50, 0], {
easing: Easing.out(Easing.cubic),
extrapolateRight: "clamp",
});

return (
<AbsoluteFill style={{ justifyContent: "center", alignItems: "center" }}>
{/* Backdrop */}
```

```
<div
style={{
position: "absolute",
inset: 0,
backgroundColor: "#000",
opacity: backdropOpacity,
pointerEvents: "none",
}}
/>
```

```
    {/* Modal */}
    <div
      style={{
        width: "500px",
        padding: "40px",
        backgroundColor: "white",
        borderRadius: "16px",
        boxShadow: "0 20px 60px rgba(0,0,0,0.2)",
        transform: `scale(${scale}) translateY(${translateY}px)`,
        textAlign: "center",
      }}
    >
      <h2 style={{ fontSize: "28px", fontWeight: "700", marginBottom: "16px" }}>
        Success! □
      </h2>
      <p style={{ fontSize: "16px", color: "#6B7280", margin: 0 }}>
        Your video has been created successfully
      </p>
    </div>
  </AbsoluteFill>
```

```
);
};
```

3.3 Advanced Transition Effects

Scene transitions that feel cinematic and premium.

```
import { interpolate, Easing, AbsoluteFill, useCurrentFrame } from "remotion";
```

```
// TRANSITION 1: Slide (Director's favorite)
export const SlideTransition: React.FC<{
from: React.ReactNode;
to: React.ReactNode;
```

```
direction?: "left" | "right" | "up" | "down";
startFrame: number;
duration: number;
}> = ({ from, to, direction = "left", startFrame, duration }) => {
const frame = useCurrentFrame();

// Direction vectors
const directionMap = {
left: { x: -1, y: 0 },
right: { x: 1, y: 0 },
up: { x: 0, y: -1 },
down: { x: 0, y: 1 },
};

const { x, y } = directionMap[direction];

// Outgoing layer position
const outX = interpolate(
frame,
[startFrame, startFrame + duration],
[0, -100 * x],
{ easing: Easing.inOut(Easing.cubic), extrapolateRight: "clamp" }
);

const outY = interpolate(
frame,
[startFrame, startFrame + duration],
[0, -100 * y],
{ easing: Easing.inOut(Easing.cubic), extrapolateRight: "clamp" }
);

// Incoming layer position
const inX = interpolate(
frame,
[startFrame, startFrame + duration],
[100 * x, 0],
{ easing: Easing.inOut(Easing.cubic), extrapolateRight: "clamp" }
);

const inY = interpolate(
frame,
[startFrame, startFrame + duration],
[100 * y, 0],
{ easing: Easing.inOut(Easing.cubic), extrapolateRight: "clamp" }
);

return (
<>
<AbsoluteFill style={{ transform: translate(${outX}%, ${outY}%) }}>
{from}
</AbsoluteFill>
<AbsoluteFill style={{ transform: translate(${inX}%, ${inY}%) }}>
{to}
</AbsoluteFill>
</>
);
};

// TRANSITION 2: Dissolve (Professional)
export const DissolveTransition: React.FC<{
from: React.ReactNode;
to: React.ReactNode;
startFrame: number;
duration: number;
}> = ({ from, to, startFrame, duration }) => {
const frame = useCurrentFrame();

const opacityFrom = interpolate(
frame,
[startFrame, startFrame + duration],
[1, 0],
{ easing: Easing.inOut(Easing.quad), extrapolateRight: "clamp" }
);

const opacityTo = interpolate(
frame,
[startFrame, startFrame + duration],
[0, 1],
{ easing: Easing.inOut(Easing.quad), extrapolateRight: "clamp" }
);

return (
<>
<AbsoluteFill style={{ opacity: opacityFrom }}>{from}</AbsoluteFill>
<AbsoluteFill style={{ opacity: opacityTo }}>{to}</AbsoluteFill>
</>
);
};

// TRANSITION 3: Circle Wipe (Modern SaaS favorite)
export const CircleWipeTransition: React.FC<{
from: React.ReactNode;
to: React.ReactNode;
startFrame: number;
duration: number;
}> = ({ from, to, startFrame, duration }) => {
const frame = useCurrentFrame();

const progress = interpolate(
frame,
[startFrame, startFrame + duration],
[0, 1],
{ easing: Easing.inOut(Easing.quad), extrapolateRight: "clamp" }
);
```

```
// Circle radius expands from center (0 to ~150% of screen diagonal)
const radius = progress * 1500;

return (
<>
{from}
<AbsoluteFill
style={{
background: "white",
clipPath: circle(${radius}px at 50% 50%),
}}
>
{to}
</AbsoluteFill>
</>
);
};
```

---

4. Performance Optimization (Critical for SaaS)

4.1 GPU Acceleration

Enable hardware acceleration for smooth rendering on complex compositions.

```
// remotion.config.ts
import { defineConfig } from "remotion";

export const config = defineConfig({
// Enable GPU acceleration for transforms, shadows, gradients, filters
enableGpuAcceleration: true,

// Server-side rendering GPU optimization
numberOfGifWorkers: 4,

// Optimal settings for SaaS videos
pixelFormat: "yuv420p", // Compatibility + file size
codec: "libx264",
crf: 18, // Quality (0-51, lower = better; 18 is excellent)
videoBitrate: "8M", // 8 Mbps for 1080p

// Framerate
fps: 30, // 24fps too cinematic, 60fps overkill for most SaaS
});
```

4.2 Performance Best Practices

```
// ✕ BAD: Creates new objects every render
export const BadComponent: React.FC = () => {
const animatedStyle = {
transform: scale(${someValue}), // Object recreation
};
return
```

```
Content
;
};

// ✅ GOOD: Static object references with dynamic values
const animatedStyleBase: CSSProperties = {
willChange: "transform",
transition: "none",
};

export const GoodComponent: React.FC = () => {
const frame = useCurrentFrame();
const scale = interpolate(frame, [0, 30], [0, 1]);

return (
<div
style={{
...animatedStyleBase,
transform: scale(${scale}),
}}
>
Content
</div>
);
};

// OPTIMIZATION: Use AbsoluteFill for fullscreen elements (native transforms)
import { AbsoluteFill } from "remotion";

export const OptimizedFullscreen: React.FC = () => {
return (
<AbsoluteFill style={{ backgroundColor: "#fff" }}>
Content fills entire canvas automatically
</AbsoluteFill>
);
};

// OPTIMIZATION: Avoid re-renders with useMemo
import { useMemo } from "react";

export const MemoizedScene: React.FC<{ items: string[] }> = ({ items }) => {
const renderedItems = useMemo(
() =>
items.map((item, i) => (
<div key={i} style={{ color: "#000" }}>
{item}
</div>
)),
[items]
);

return
{renderedItems}
```

```
;
};
```

4.3 Rendering Optimization Commands

Fast local preview (lower quality for speed)

```
npm run start
```

Render with GPU acceleration

```
npx remotion render src/Root.tsx SaaS_Explainer_60s out.mp4
--enable-gpu-acceleration
```

Batch render multiple compositions

```
npx remotion render src/Root.tsx --enable-gpu-acceleration
```

Render specific resolution for different platforms

Instagram (1:1)

```
npx remotion render src/Root.tsx Video_Insta video-insta.mp4
--width 1080 --height 1080
```

YouTube (16:9)

```
npx remotion render src/Root.tsx Video_Youtube video-youtube.mp4
--width 1920 --height 1080
```

LinkedIn (4:5)

```
npx remotion render src/Root.tsx Video_LinkedIn video-linkedin.mp4
--width 1080 --height 1350
```

---

5. Complete SaaS Video Example

Full working example: Professional 60-second explainer.

```
// src/components/scenes/Complete60sExplainer.tsx
import React from "react";
import {
Sequence,
AbsoluteFill,
useCurrentFrame,
interpolate,
Easing,
} from "remotion";
import { AnimatedHeading } from "../common/AnimatedText";
import { StaggeredFeatureList } from "../common/StaggeredList";

interface Complete60sProps {
productName: string;
tagline: string;
features: string[];
ctaText: string;
accentColor: string;
}

export const Complete60sExplainer: React.FC<Complete60sProps> = ({
productName,
tagline,
features,
ctaText,
accentColor,
}) => {
return (
<AbsoluteFill style={{ backgroundColor: "#FFFFFF" }}>
{/* SCENE 1: Intro (0-10s = 0-300 frames) */}
```

```
{/* SCENE 2: Problem (10-25s = 300-750 frames) */}
<Sequence from={300} durationInFrames={450}>
  <ProblemScene tagline={tagline} color={accentColor} />
</Sequence>

{/* SCENE 3: Solution (25-45s = 750-1350 frames) */}
<Sequence from={750} durationInFrames={600}>
  <SolutionScene
    features={features}
    color={accentColor}
    productName={productName}
  />
</Sequence>

{/* SCENE 4: CTA (45-60s = 1350-1800 frames) */}
<Sequence from={1350} durationInFrames={450}>
  <CTAScene ctaText={ctaText} color={accentColor} />
```

```
    </Sequence>
  </AbsoluteFill>
```

```
);
};

// SCENE 1 DETAIL
const IntroScene: React.FC<{ productName: string; color: string }> = ({
productName,
color,
}) => {
const frame = useCurrentFrame();

// Animated background gradient
const gradientAngle = interpolate(frame, [0, 300], [0, 45], {
easing: Easing.linear,
});

// Logo scale
const logoScale = interpolate(frame, [30, 60], [0.5, 1], {
easing: Easing.out(Easing.elastic(1.2)),
extrapolateRight: "clamp",
});

// Text stagger
const headingOpacity = interpolate(frame, [60, 90], [0, 1], {
easing: Easing.out(Easing.quad),
extrapolateRight: "clamp",
});

const taglineOpacity = interpolate(frame, [120, 150], [0, 1], {
easing: Easing.out(Easing.quad),
extrapolateRight: "clamp",
});

return (
<AbsoluteFill
style={{
background: linear-gradient(${gradientAngle}deg, ${color}15, ${color}05),
display: "flex",
flexDirection: "column",
justifyContent: "center",
alignItems: "center",
gap: "30px",
}}
>
{/* Logo */}
<div
style={{
fontSize: "80px",
transform: scale(${logoScale}),
transformOrigin: "center",
}}
>
✨
</div>
```

```
    {/* Main heading */}
    <h1
      style={{
        fontSize: "64px",
        fontWeight: "800",
        color: "#000",
        margin: 0,
        opacity: headingOpacity,
        textAlign: "center",
        maxWidth: "900px",
      }}
    >
      Introducing {productName}
    </h1>

    {/* Tagline */}
    <p
      style={{
        fontSize: "28px",
        color: color,
        fontWeight: "600",
        margin: 0,
        opacity: taglineOpacity,
        maxWidth: "800px",
        textAlign: "center",
      }}
    >
      The smarter way to work
    </p>
  </AbsoluteFill>
```

```
);
};
```

```
// SCENE 2 DETAIL
const ProblemScene: React.FC<{ tagline: string; color: string }> = ({
tagline,
color,
}) => {
const frame = useCurrentFrame();

const scaleIn = interpolate(frame, [0, 40], [0.8, 1], {
easing: Easing.out(Easing.cubic),
extrapolateRight: "clamp",
});

return (
<AbsoluteFill
style={{
display: "flex",
flexDirection: "column",
justifyContent: "center",
alignItems: "center",
gap: "40px",
padding: "60px",
}}
>
<div
style={{
fontSize: "120px",
transform: scale(${scaleIn}),
transformOrigin: "center",
}}
>
▯
</div>
```

```
<h2
  style={{
    fontSize: "56px",
    fontWeight: "700",
    color: "#000",
    margin: 0,
    textAlign: "center",
    maxWidth: "900px",
  }}
>
  {tagline}
</h2>
```

```
    <p
      style={{
        fontSize: "24px",
        color: "#666",
        margin: 0,
        textAlign: "center",
        maxWidth: "800px",
        fontWeight: "400",
      }}
    >
      Teams waste hours on repetitive, manual workflows that don't scale.
    </p>
  </AbsoluteFill>
```

);
};

// SCENE 3 DETAIL - Features
const SolutionScene: React.FC<{
features: string[];
color: string;
productName: string;
}> = ({ features, color, productName }) => {
return (
<AbsoluteFill
style={{
display: "flex",
flexDirection: "column",
justifyContent: "center",
alignItems: "center",
padding: "60px",
gap: "40px",
}}
>
<h2
style={{
fontSize: "48px",
fontWeight: "700",
color,
margin: "0 0 20px 0",
}}
>
{productName} solves this with:
</h2>

```
    <div style={{ width: "100%", maxWidth: "800px" }}>
      <StaggeredFeatureList
        items={features}
        staggerDelay={20}
        startFrame={60}
        itemDuration={60}
      />
    </div>
  </AbsoluteFill>
```

```
);
};

// SCENE 4 DETAIL - CTA
const CTAScene: React.FC<{ ctaText: string; color: string }> = ({
ctaText,
color,
}) => {
const frame = useCurrentFrame();

const buttonScale = interpolate(frame, [100, 150], [0.7, 1], {
easing: Easing.out(Easing.elastic(1.15)),
extrapolateRight: "clamp",
});

const textOpacity = interpolate(frame, [0, 30], [0, 1], {
easing: Easing.out(Easing.quad),
extrapolateRight: "clamp",
});

return (
<AbsoluteFill
style={{
display: "flex",
flexDirection: "column",
justifyContent: "center",
alignItems: "center",
gap: "40px",
background: linear-gradient(135deg, ${color}20, ${color}05),
}}
>
<h2
style={{
fontSize: "56px",
fontWeight: "700",
color: "#000",
margin: 0,
```

textAlign: "center",
opacity: textOpacity,
}}
>
Ready to transform your workflow?
</h2>

```
    <button
      style={{
        backgroundColor: color,
        color: "white",
        padding: "20px 60px",
        fontSize: "24px",
        fontWeight: "600",
        border: "none",
        borderRadius: "12px",
        cursor: "pointer",
        transform: `scale(${buttonScale})`,
        transformOrigin: "center",
        boxShadow: `0 12px 40px ${color}40`,
        transition: "none",
      }}
    >
      {ctaText}
    </button>

    <p
      style={{
        fontSize: "18px",
        color: "#666",
        margin: 0,
        opacity: textOpacity,
      }}
    >
      Free trial. No credit card required.
    </p>
  </AbsoluteFill>
```

```
);
};
```

---

6. Top-Tier SaaS Video Examples: What Makes Them Work

Brand	Key Technique	Why It Works
Calendly	Minimalist design + smooth transitions	Clean, non-overwhelming, focuses on benefit
Superhuman	Problem-first + relatable narrative	Emotional connection before solution
RingCentral	3D UI + photorealistic animations	Premium perception, technical credibility
Loom	Human-centric + quick pacing	Authentic feel, respects viewer attention
Slack	Feature highlights with interactive demos	Concrete value + interaction proof
Duolingo	Gamified animations + personality	Engaging, memorable, differentiating

Key Principles from Top Examples:

1. **Problem-First Narrative** - Lead with audience pain, then solution
2. **Smooth 60-90 second runtime** - Golden zone for engagement
3. **Spring animations for premium feel** - Never robotic, always responsive
4. **Staggered feature reveals** - Let audience digest each point
5. **Strong visual hierarchy** - One clear focal point per scene
6. **Consistent brand color usage** - 2-3 colors max
7. **Professional typography** - Sans-serif, clear sizing hierarchy
8. **Subtle sound design cues** - Animation beats sync with audio
9. **Clear CTAs** - Never ambiguous about next step
10. **Mobile-first thinking** - Test on phone (many viewers)

---

7. Advanced: Parametrized Video Generation

Create multiple variations dynamically.

```
// Generate 10 product demos with different features automatically
const PRODUCTS = [
{
name: "Analytics Pro",
tagline: "Track metrics that matter",
features: ["Real-time dashboards", "Custom reports", "AI insights"],
color: "#3B82F6",
},
{
name: "Marketing Automation",
tagline: "Scale your campaigns",
features: ["Email sequences", "Lead scoring", "A/B testing"],
color: "#EC4899",
},
// ... more products
];

// Root.tsx: Auto-generate compositions
export const RemotionRoot: React.FC = () => {
return (
<>
{PRODUCTS.map((product) => (
<Composition
key={product.name}
id={${product.name.replace(/ /g, "_")}_Video}
component={Complete60sExplainer}
durationInFrames={1800}
fps={30}
width={1920}
height={1080}
defaultProps={product}
/>
))}
</>
);
};

// CLI: Render all variations
// npx remotion render --concurrency=4
```

8. Troubleshooting & Common Pitfalls

Issue	Cause	Solution
Janky animations	Missing willChange: "transform"	Add to animated elements
Slow render time	GPU not enabled	Use --enable-gpu-acceleration
Text looks blurry	Rasterization	Use SVG text or system fonts
Spring overshoots	overshootClamping false	Set to true if unwanted
Stagger timing off	Incorrect frame math	Double-check: startFrame + (index × stagger)
Rendering fails locally	FFMPEG missing	npm install ffmpeg or use server
Color spaces mismatched	YUV codec issue	Stick to yuv420p for web delivery

---

9. Resources & Community

- **Official Docs:** https://www.remotion.dev/docs
- **Discord Community:** https://remotion.dev/discord
- **GitHub Examples:** https://github.com/remotion-dev/remotion
- **Interactive Playground:** https://www.remotion.dev/docs/layout-utils

---

10. Checklist: Before Publishing

- [ ] Aspect ratio tested (1080p, 4:5, 9:16)
- [ ] All animations smooth at 30fps
- [ ] Typography readable at 1080p minimum
- [ ] Colors accessible (WCAG AA contrast minimum)
- [ ] CTA clear and clickable
- [ ] Video under 100MB for web
- [ ] Rendered with GPU acceleration
- [ ] Audio synced (if applicable)
- [ ] Mobile preview verified
- [ ] Composition tested in Claude Code